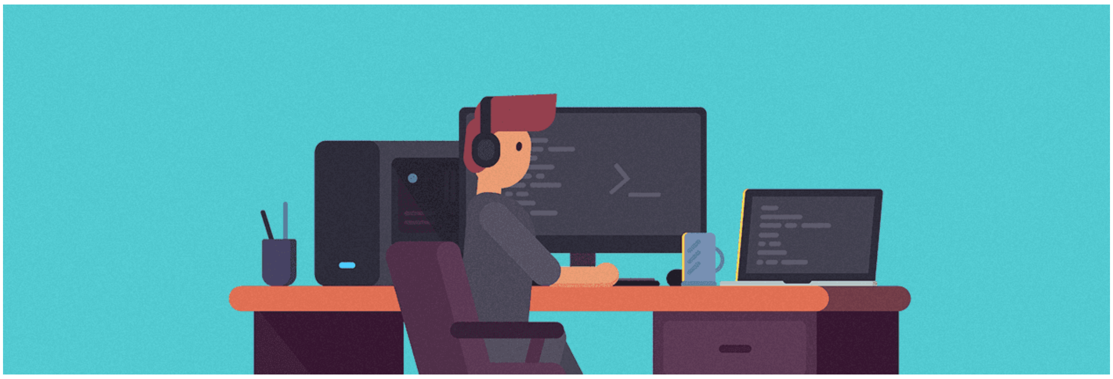


06 月 07 日

# 第七章：用 Ulysses 搞定办公文档

| 本文为付费栏目文章，您已订阅，可阅读全文 |

想让一位习惯了用 Word 来处理办公文档的用户转用 Ulysses 作为写作工具，要迈过两道坎：

1. 熟悉 Markdown 语法；
2. 了解如何在 Ulysses 中处理文档的排版。

经过前面几章的介绍，我想各位已经基本掌握了在 Ulysses 中写 Markdown 的方法和技巧，坚持一段时间的使用，就会顺利迈过第一道坎。但是多数人没能完成从 Word 到 Ulysses 的转换，往往是被难在了第二道坎。

严格来说，**Ulysses 作为一款写作工具**，不应该被视作 Word 的替代品。但是在实际使用中，不少用户习惯在一款软件中，完成从写作到排版的流程。于是选择忍受 Word 效率不高的写作体验，放弃了使用 Ulysses 来处理工作文档。

而提到调整文章的排版，自然会想到自定义样式。

在第五、六章中，我们已经见识了通过 CSS 文件自定义样式，来满足对 HTML 和 ePub 的排版需求。而在办公环境中，PDF、DOCX 格式的文档才是主流。此时就可以借助 Ulysses Style Sheets（以下简称 USS ），帮助我们导出满足工作需要的文档，在 Ulysses 中，**尽可能多，甚至完全解决文档的排版需求**。

## 办公，该用 Ulysses 做什么

在应用内部和 Ulysses Style Sheets 中，已经有不少 PDF 和 DOCX 的样式，在对文档排版要求不高时，可以直接使用。但是真正在职场工作过的人就知道，工作文档的排版美观度不占第一优先位置，而是必须根据公司要求，保持文档的版式一致。

除了对版式一致性的要求，办公文档还必须考虑图片、表格等元素，以及封面目录等。Ulysses 作为一款写作工具，显然是很难全部满足所有需求的。但是要是写完文稿后直接导出成 DOCX，再放到 Word 中去完成，又没有发挥 USS 的便利。

当我们讨论用 Ulysses 来处理办公文档时，就必须先谈论排版流程中，Ulysses 能在哪些步骤发挥作用。

## 办公文档的排版流程

尽管每个人处理文档的习惯有所不同，但想对一份文档进行排版，有些步骤是必不可少的。

1. **设计文档页面的布局**。具体来说就是页面的页眉页脚、页码脚注的位置和样式，以及文档宽度等参数；
2. **确定文档内容**。文章中的标题、图片、表格等等内容，应该在调整格式前就完全确定；
3. **设置文稿格式**。设计字、段落、图片等内容的格式；

4. **创建目录和索引**。对于稍长的文档来说，目录通常是必不可少的。而有时也会根据需要，加上关键词或者图片索引。
5. **添加其他页面**。在第一步中只是设置了页眉页脚的风格，现在就要设定好它们的内容。其他需要添加的还有封面、附录等必要的内容。

经过这样五步的处理，一份合格的文档就基本排版完成。

## Ulysses 应该如何发挥作用

再一次强调，Ulysses 是一款**写作工具**，和 Word 既不是竞争关系，也不是想要取代它。事实上两者应该是互相合作，Ulysses 负责写作，Word 负责排版。也就是说，其实上述排版流程中的第二步，才应该是 Ulysses 应该发挥作用的地方。

不过得益于 USS 的高度自定义特性，让我们**在 Ulysses 中就能完成前三步的排版工作**。

排版第一步对页面布局的设定，就可以在 Ulysses 中完成。在 Ulysses 中已有的样式，其实都经过了这一步，只不过我们直接预览和导出，没有手动干预。在 USS 文件中，这些设定是必不可少的。

第二步则是 Ulysses 的强项。Markdown 的优势就在于利用简单的语法标记，去书写标题、引用、代码等多种文本元素。而在导出时也是完全不需要再手动修改，可以直接得到一份结构完整的文稿。

在第二步中写好了文稿，USS 「捎带手」就完成了对它们样式的设定。不论是标题、正文的字体字号，还是段前段后的缩进，都可以在 USS 进行自定义。

当然 Ulysses 作为写作工具，在排版上还是有所局限。

第四、第五步中要求其实超出了对文字的处理。例如生成目录，虽然 Markdown 可以通过其他工具创建目录，但其实难点在于它如何根据文档内容自动更新以及样式设计。至于封面设计等任务，其实都不是适合在 Ulysses 中完成。

所以我们将注意力集中在前三步，看看 USS 如何帮助我们，快速搞定这些办公文档的排版需求。不过在这之前，我们先来初步了解 USS。

## 初步入门 Ulysses Style Sheets（USS）

如果你对第五、六章的内容有所理解，会很清楚地认识到在 Ulysses 中排版的秘密：**将富文本编辑器中图形化操作，转换为纯文本格式的样式文件**。处理 HTML 和 ePub 文件时我们用的是 CSS 文件。而处理 PDF 和 DOCX 文件时，就要用到 Ulysses Style Sheets。

### USS 与 CSS 的相同之处

先给各位吃一颗定心丸：**如果你已经对 CSS 有所了解，那么看完本文的介绍，你就会明白 USS 该怎么用了。**

Ulysses Style Sheets 是 Ulysses 官方定义的一套处理 PDF 和 DOCX 文件的样式文件格式。虽然没有明确说它参考了 CSS 或是 CSS 的子集，但从实质上来看，**它的语法结构和使用方法，与 CSS 如出一辙**。

和学习 CSS 时一样，我建议各位拷贝一份 Ulysses 内置的样式文件作为参考，下文以 Papers 样式为例。

拷贝 Papers

首先 **USS 和 CSS 的基本语法结构是一样的**。打开 `Papers.ulsss` 文件后，可以看到，和 CSS 类似，USS 也是由：

```
heading-1 {
    style-title:            "Heading 1"
    text-alignment:         center
    font-weight:            bold
}
```

这样一块块的内容组成。这其实就是 CSS 中的结构：

```
选择器{
    属性: 值
    ...
}
```

其次 **USS 和 CSS 的作用机制也是一样的**。在 USS 中，你不仅需要对标题、粗体、引用等等元素进行样式设定，还要设定文章的主体等等参数。但实际上你可能已经发现，类似的工作我们在自定义 CSS 样式的时候就已经做过。

## USS 与 CSS 的不同之处

从本质上来说，USS 完全可以看做针对 Ulysses 的方言版 CSS。但之所以要这么干，而不是直接使用 CSS 来生成 PDF，USS 自然和 CSS 也有所不同。

最明显的不同是**选择器名称的不同**。 一级标题在 USS 中是 `heading-1`，而在 CSS 通常是 `h1`。又譬如对文档整体的设置，USS 中是 `document-settings`，而 CSS 中是 `body`。在第六章我们看到过 CSS 中的选择器，在 USS 中都有对应，但都是不同的名字。

当然 USS 并不是为了不同而故意改变 CSS 选择器的名称。CSS 的出现是为了解决网页样式的问题，在使用上非常灵活。如果你打开少数派官网的 CSS 文件，会发现它的选择器名称也与常见的 CSS 不一样。事实上为了开发上的方便，自定义 CSS 文件中的选择器名称是很常见的现象。

但正是 CSS 的灵活就导致了它与 USS 的第二个不同：**应用的范围不同**。USS 的出现与 CSS 不同，**它专为在 Ulysses 中优雅地导出 PDF 文件（和 DOCX）**。为了保证样式文件的可用性，USS 对选择器、属性和值的种类都进行了限制。这样所有人写出来的样式文件，都能被 Ulysses 正确地解析。

或许对于 CSS 专家来说，重新定义一种与 CSS 类似的样式文件显得意义不大，因为它本身足够强大灵活。但对于普通用户来说，USS 的出现让学习范围变的清晰，更易于普通用户掌握在 Ulysses 中排版。

## 探索 Ulysses Style Sheets

因为 USS 是 Ulysses 官方定义的样式格式，故而在官方参考文档 Ulysses Style Sheets Reference 中，列出了所有的选择器类型、属性种类和对应的值。

有了 CSS 的学习经验，我们不再需要从头介绍选择器、属性与值的概念与用法。接下来对 USS 中出现的重要术语，进行快速而全面的概览。

### Style Selectors —— 样式选择器

这是 USS 中最常见的选择器，它**用来定义文档中文稿的样式**，在 CSS 中对应的就是元素选择器。

在 USS 中，样式选择器被细分为两种 ：

1. 定义选择器（Definitions Selector）

2. 通用选择器（Generic Selector）

其实两者本质上没有差别，只是前一种**用来设定具体的元素**，例如一级标题 `heading-1`、有序列表 `list-ordered`。而后者**用来设定一组同类别的元素**，例如所有的标题 `heading-all`、所有种类的列表 `list-all`。

而在书写语法上，样式选择器是最标准格式。例如 Papers 中的选择器 `document-settings`：

```
document-settings {
    page-inset-top:     1in;    page-inset-inner:   1in
    page-inset-bottom:  1in;    page-inset-outer:   1in

    two-sided:              no
    page-binding:           left

    footnote-enumeration:   per-section
    footnote-placement:     end-of-page
    footnote-style:         decimal
}
```

虽然内容多，但是还是最基础的格式：

```
选择器{
    属性: 值
    ...
}
```

和 CSS 一样，一对属性和值组成一条声明。不过不同的是，在 **USS 中每条声明间不需要用分号隔开**。只有当你想把多条声明写在一行时，才需要用分号。

## Relative Selectors —— 关联选择器

关联选择器并不是一类全新的选择器，而是**将两种样式选择器组合起来，在特定的场景发挥作用**。

在第六章中我们介绍过后代选择器，它可以用来定义在代码块中的文字、或是引用中的粗体这类受两种选择器共同影响的内容，其实它就是一种关联选择器，在 CSS 中被称为派生选择器。

其实在 CSS 中派生选择器分为很多种，之前限于篇幅和内容限制没有完全展开介绍。而在 USS 中，共有三种关联选择器：

1. 后代选择器：ParentDefinition ChildDefinition
2. 相邻兄弟选择器：ParentDefinition + ChildDefinition
3. 子元素选择器：ParentDefinition > ChildDefinition

后代选择器我们在第六章已经介绍过，那么同为关联选择器，相邻兄弟选择器和子元素选择器分别负责干什么呢？

**相邻兄弟选择器**

通常标题下的段落会离得与标题稍远一些，以突出标题。但是这在标题或段落的样式选择器中是无法调整的，它们只能控制各自统一的段前段后。这时相邻兄弟选择器就可以发挥作用。

组成相邻兄弟的两个选择器，顾名思义要**是平级（兄弟）关系**，而不是像后代选择器中一样，一个嵌套在另一个元素中。在调整标题与段落间距的场景中，可以认为标题与段落同属于文档整体（`body`），故而是平级关系。

所以就可以将一级标题和段落的样式选择器组合起来，写成：

```
heading-1 + paragraph {
    margin-top: 10pt
}
```

其中 `heading-1` 是一级标题，`paragraph` 是段落的样式选择器。必须注意的是，这两个元素在文中必须是连续出现的，中间不能夹杂其他元素。而且只会对兄选择器 `heading-1` 后的第一个弟选择器 `paragraph` 起作用。

**子元素选择器**

再来看看子元素选择器。

假设你通过后代选择器设定了列表中的粗体。有一份列表中又嵌套了一份列表，而在这份被嵌套的列表中，你不希望出现粗体效果，后代选择器就不再适用，因为出现在列表中的所有粗体，都会被后代选择器设定。

而子元素选择器就可以解决这个问题。同后代选择器一样，**子元素选择器的两个样式选择器之间也是嵌套的关系**。但是不同的是，**嵌套的元素只能是被嵌套元素的下一级**，而不能是子级的子级。用家族关系来对应就是，后代选择器只规定两者是长辈与后辈，至于是父子关系，还是祖孙关系不影响使用。而子元素选择者严格限定两者是父子关系。

从 USS 来表达就是：

```
orderedList > inline-strong {
    font-size: 15pt
}
```

可以看出，子元素选择器其实对后代选择器进行了更严格的约束。

在 USS 中，还有伪类、继承等特性需要掌握。在后面结合实例再做讲解。

## 创建一份适合办公文档的样式

在第一节中，我们梳理了办公文档的排版流程，并且明确了 Ulysses 应该起到的作用。接下来就进入实战阶段，结合实际的排版需求，来介绍**如何创建一份适合办公文档的 USS 样式文件**。

需要说明的是，办公文档的排版很难找出最佳设计，而是根据公司要求不同有所调整。所以下文不会强调地告诉你该把字体调多大，行距选择多少。而是将重点放在对选择器和重要属性的介绍上。

为了方便大家编辑和学习，我们就在 `papers.ulss` 的基础上来进行编辑。

### 设计文档页面布局

为了设置文档的整体页面布局，需要逐个调整三类选择器：

1. `document-settings`：用来设定整体文档风格，例如分栏、页面宽度等等；
2. `Headers and footers`：其中包含两个选择器，`area-header` 和 `area-footer`，分别用来设定页眉页脚的位置和风格；

3. `area-footnotes`：用来设定文稿内脚注的风格。

### document-settings

```
document-settings {
page-inset-top:     1in;    page-inset-inner:   1in
page-inset-bottom:  1in;    page-inset-outer:   1in

two-sided:              no
page-binding:           left

footnote-enumeration:   per-section
footnote-placement:     end-of-page
footnote-style:         decimal
}
```

在 `Papers.ulss` 中，首先**对版心的位置进行进行了设定**。和 CSS 的盒子模型一样，top、bottom、inner 和 outer 其实就对应版心与上下左右边界相隔的距离。在 Papers 中，他们都被设定为一英寸。

其次文档被设定为单面打印（`two-side`），并且在文档左侧装订（`page-binding: left`）。

看到最后三行可能你会有些疑惑：为什么在这里出现了对脚注的设定，而不是 `area-footnotes` 选择器中？其实这里是对脚注进行内容和位置的设定，而 `area-footnotes` 中是对脚注样式上的设定。

除了已有的属性，可以在参考文档中查看所有可用的属性。其中特别值得注意的是分节符 `section-break` 这个属性。

这里的「节」并不是指文章的章节，而是将文档分为不同的部分，例如序言、正文、附录等等。每个部分可以重现开始页码和脚注的计数。例如将属性 `page-number-reset` 设定为 `per-section`，每节就是重现开始计数页码。

分节符还在 PDF 打印中起作用。在双面打印时，就要避免节的第一页被打印在偶数页。如果遇上这种情况，插入分节符就可以通过在偶数页插入空白页的方式，确保每节的第一页打印在奇数页。

在 USS 中可以选择标题和段落作为分节符。需要注意的是，当你设定了某一级标题作为分节符，这一级和比它高一级的标题都会被当作分节符。例如设定二级标题为分节符，实际上一级标题和二级标题都会作为分节符。

### Headers and footers

对页眉和页脚的设定，分为 `area-header` 和 `area-footer` 两个选择器。其中 Papers 只对页眉进行了设定：

```
area-header {
    content:                page-number
    text-alignment:          right
    top-spacing:             0.35in
}
```

`area-header` 和 `area-footer` 可以设定据上下边的间距，当然最重要的还是 `content` 属性，可以设定为页码（`page-number`）或者当前节标题（`heading`）。

在对页眉页脚的设定中，还可以使用伪类，还更加灵活地进行设定。USS 中的伪类和 CSS 中的概念用法都完全一致。在 `area-header` 和 `area-footer` 中，你可以使用三种伪类：

1. `:first-page`：只在每节第一页设定页眉页脚；
2. `:left-page`：（双面打印时）只在左侧页面（奇数页）设定页眉页脚；
3. `:right-page`：（双面打印时）只在右侧页面（偶数页）设定页眉页脚。

经常在办公文档中，每节的第一页是尽量保持页面整洁而不设定页眉页脚，于是可以借助伪类实现这种效果：

```
area-header :first-page {
    content:          none
}
```

```
area-footer :first-page {
    content:          none
}
```

### area-footnotes

对脚注区域的设置，可以通过 `area-footnotes` 完成。通过观察参考文档，可以发现需要设定的其实就是三类：

1. 对脚注编号的设定：`anchor-alignment` 和 `anchor-inset`。
2. 对正文和脚注之间的分割线的设定：`divider-length`、`divider-position` 、`divider-spacing` 和 `divider-width`。
3. 对脚注内容的格式设定：`text-inset` 和 `top-spacing`。

事实上，除了这些列出来参数，一般常见对字体的属性也可以写在 `area-footnotes` 之中，以调整脚注的格式，例如在 Papers 中：

```
area-footnotes {
    divider-length:        10cm
    divider-spacing:       0pt
    divider-width:         0.7pt
    anchor-alignment:      left
    anchor-inset:          4cm
    margin-left:           -2em
}
```

这些参数都需要你根据公司对文档的规范进行设定。

## 确定文档内容

设置完文章的基本布局，接下里就是确定文档的内容。具体来说，就是**设定好文章中的标题、段落、列表等元素**。

可能你会奇怪，在 Ulysses 写作时，这些不就是本来应该完成的吗？的确，在写作时顺手完成这部分的排版任务，这正是 Markdown 的魔力。在 Word 中，输入的文字默认都是普通文本，需要手动指定文本的种类，方便之后在样式窗格中自动修改样式。

所以在 Ulysses 中，我们基本跳过这步，因为除了视频，Ulysses 中的语法标记基本能正常转换，并不需要我们做额外的设置。

## 设置文稿格式

终于到了各位最熟悉的一步！在这一步中，我们重点研究如何将文档各部分的样式，用 USS 设定好。

### 设置图片的样式

之所以要最先介绍对图片的处理，不但是因为它是办公文档中最必不可少的元素，也是在从纯文本的 Markdown 转换到富文本文档时最难处理的对象。

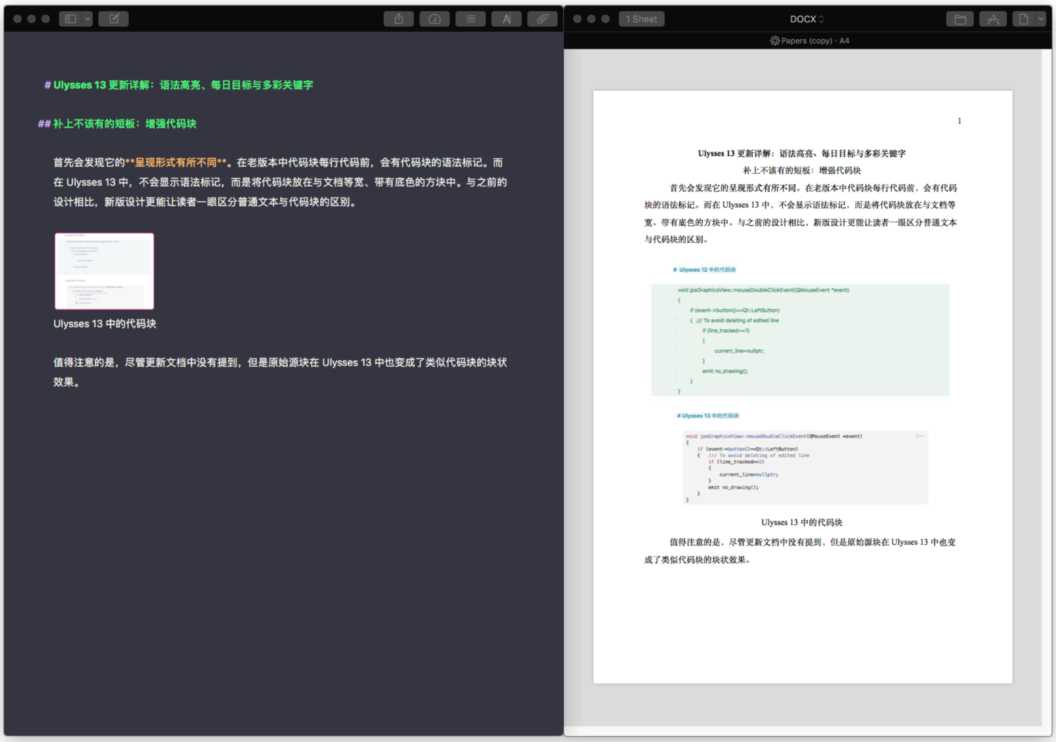
对于图片首先**要解决的是图片说明在 PDF 和 DOCX 中不导出的问题**。通常在 Ulysses 中，可以在图片弹窗中添加说明，在导出为 HTML 时，就会转换为图片下部的说明。但是在转换成 PDF 和 DOCX 时，就不会出现这样的效果。

如果还要在导出的 DOCX 中，再手动添加图片描述，不免太过麻烦。其实解决方案已经隐藏在 Papers 样式中：

```
paragraph-figure + paragraph {
   text-alignment:        center
   first-line-indent:      0pt
   margin-bottom:         2.75pt
}
```

由 `paragraph-figure` 和 `paragraph` 两个样式选择器组成了一个相邻兄弟选择器。前者用来设定图片，后者用来设定段落，两者结合起来，就可以设定图片后的第一个段落。

而在具体的属性设置中，将文字设置为了居中，去掉了段落的首行缩进，并且将段后设定为 2.75pt。这样设置的效果，就将图片后的第一个段落，摆在了图片说明的位置，达到了想要的效果。



# Ulysses 13 更新详解：语法高亮、每日目标与多彩关键字

## 补上不该有的短板：增强代码块

首先会发现它的**呈现形式有所不同**。在老版本中代码块每行代码前，会有代码块的语法标记。而在 Ulysses 13 中，不会显示语法标记，而是将代码块放在与文档等宽、带有底色的方块中。与之前的设计相比，新版设计更能让读者一眼区分普通文本与代码块的区别。

Ulysses 13 中的代码块

值得注意的是，尽管更新文档中没有提到，但是原始源块在 Ulysses 13 中也变成了类似代码块的块状效果。

Ulysses 13 更新详解：语法高亮、每日目标与多彩关键字

补上不该有的短板：增强代码块

首先会发现它的呈现形式有所不同。在老版本中代码块每行代码前，会有代码块的语法标记。而在 Ulysses 13 中，不会显示语法标记，而是将代码块放在与文档等宽、带有底色的方块中。与之前的设计相比，新版设计更能让读者一眼区分普通文本与代码块的区别。

# Ulysses 12 中的代码块

```
void jpsGraphicsView::mouseDoubleClickEvent(QMouseEvent *event)
{
    if (event->button()==Qt::LeftButton)
    {  /// To avoid deleting of edited line
        if (line_tracked==1)
        {
            current_line=nullptr;
        }
        emit no_drawing();
    }
}
```

# Ulysses 13 中的代码块

```
void jpsGraphicsView::mouseDoubleClickEvent(QMouseEvent *event)
{
    if (event->button()==Qt::LeftButton)
    {  /// To avoid deleting of edited line
        if (line_tracked==1)
        {
            current_line=nullptr;
        }
        emit no_drawing();
    }
}
```

Ulysses 13 中的代码块

值得注意的是，尽管更新文档中没有提到，但是原始源块在 Ulysses 13 中也变成了类似代码块的块状效果。

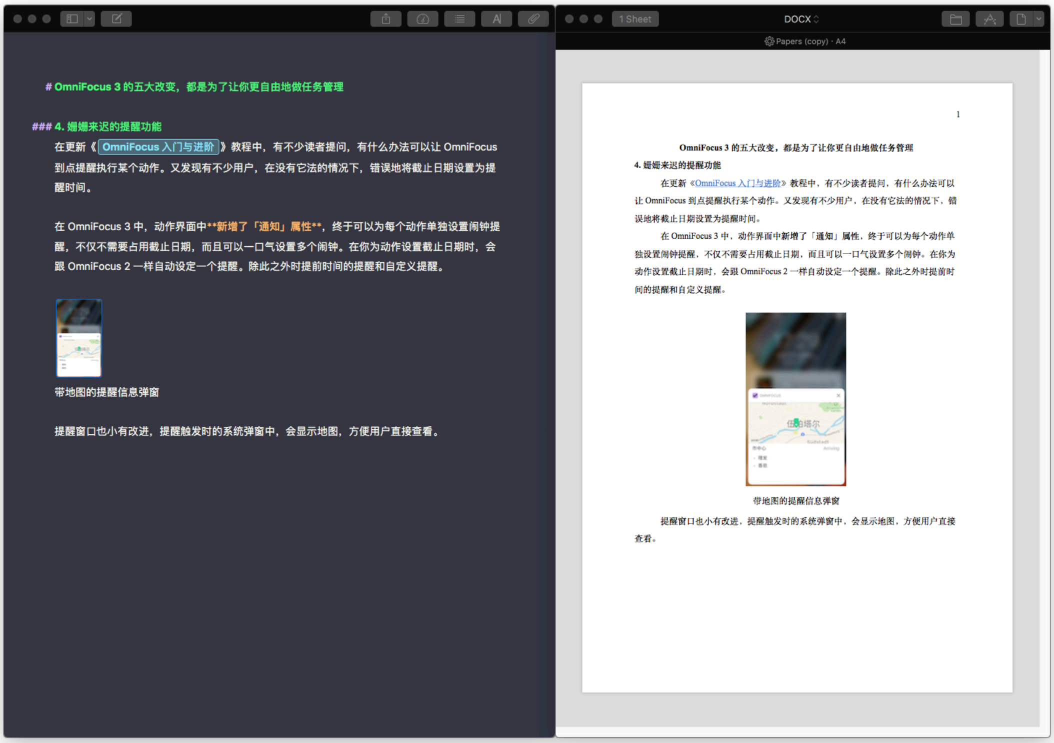
## 通过兄弟选择器设定图片说明

不过必须注意的是，此时图片说明的文本仍属于段落文本，只是摆的位置像是图片说明，并不真的类似 Word 中对图片的题注。生成真正的图片说明的方法，会在下一章介绍另一种工具帮助我们完成。

其次我们来解决图片居中的问题。在上一章中我们介绍了 CSS 文件中图片居中的办法，在 USS 中思路是一样的，但在 USS 中更加方便：

```
paragraph-figure {
    margin-top:     5mm
    margin-bottom:      5mm
    text-alignment:         center
}
```

在 `paragraph-figure` 中，直接使用 `text-alignment` 属性。虽然看名字它是用来调整文本段落的对齐方式的，但是经我测试，它也是可以用于图片的居中。



```
# OmniFocus 3 的五大改变，都是为了让你更自由地做任务管理

### 4. 姗姗来迟的提醒功能
在更新《OmniFocus 入门与进阶》教程中，有不少读者提问，有什么办法可以让 OmniFocus 到点提醒执行某个动作。又发现有不少用户，在没有它法的情况下，错误地将截止日期设置为提醒时间。

在 OmniFocus 3 中，动作界面中**新增了「通知」属性**，终于可以为每个动作单独设置闹钟提醒，不仅不需要占用截止日期，而且可以一口气设置多个闹钟。在你为动作设置截止日期时，会跟 OmniFocus 2 一样自动设定一个提醒。除此之外时提前时间的提醒和自定义提醒。

带地图的提醒信息弹窗

提醒窗口也小有改进，提醒触发时的系统弹窗中，会显示地图，方便用户直接查看。
```

1

**OmniFocus 3 的五大改变，都是为了让你更自由地做任务管理**

**4. 姗姗来迟的提醒功能**

在更新《OmniFocus 入门与进阶》教程中，有不少读者提问，有什么办法可以让 OmniFocus 到点提醒执行某个动作。又发现有不少用户，在没有它法的情况下，错误地将截止日期设置为提醒时间。

在 OmniFocus 3 中，动作界面中新增了「通知」属性，终于可以为每个动作单独设置闹钟提醒，不仅不需要占用截止日期，而且可以一口气设置多个闹钟。在你为动作设置截止日期时，会跟 OmniFocus 2 一样自动设定一个提醒。除此之外时提前时间的提醒和自定义提醒。

带地图的提醒信息弹窗

提醒窗口也小有改进，提醒触发时的系统弹窗中，会显示地图，方便用户直接查看。

图片居中的效果

Tip：只要不超过文档的宽度，在 USS 中放在同一行的图片是可以并排的。

遗憾的是，作为一款纯文字编辑器，对图片的处理有着天然的不足，在 USS 中是没有办法像在 Word 中一样调整图片等显示效果。相比 CSS，在 USS 中也没法设置图片浮动。如果有这方面的需求，只能导出为 DOCX 后在 Word 中进行调整。

### 设置文档主体、段落与标题的样式

接下来我们来看看 Papers 中对文档主体、段落和标题的设定：

```
defaults {
    font-family:           "Times New Roman"
    font-size:             $font-size
    line-height:           $line-height

    text-alignment:        left
    hyphenation:           no
}

paragraph {
    style-title:           "Paragraph"
    first-line-indent:     $indent
}

heading-all {
    keep-with-following:   true
    text-alignment:        left
}

heading-1 {
    style-title:           "Heading 1"
    text-alignment:         center
    font-weight:           bold
}

// 以下还有各级标题的设置，不一一列出
```

在对各类文章元素进行样式设定之前，必须**先设定一个基础样式**，然后在此基础上进行自定义。在 CSS 文件中这个基础样式是 `body`，而在 USS 则是 `defaults`。

在参考文档中你可以看到所有可用的属性。首先要设定的当然是字体字号。可以看到默认是 `"Times New Roman"`，你可以根据需要修改为华文黑体（ST Heiti）或者宋体（Song）。

Tip：在这个维基百科页面中，可以查看 macOS 字体的英文名。

而在设定字号和行高时，发现了它不是直接给出值，而是给出了**变量**。

在 `Papers.ulsss` 中可以看到：

```
$mark-color =                #FEFECC
$black =                    #000000
$light-grey =                #cccccc
$cool-blue =                #3875d7

$indent =                   0.5in
```

```
$font-size =                12pt
$line-height =              24.5pt
```

这些变量被提前设定。在整个 USS 文件中，你可以随时使用这些变量，而不用每次都输入一遍。

除了减少输入的次数，使用变量的另一个好处就是它可以通过**表达式**进行计算。例如变量 `$font-size` 被设定为 12pt，在 `defaults` 中被当做基础样式的字号。当我们再给一级标题设定字号时，就不用写出具体的 16pt 或者 18 pt，而是可以直接写成 `$font-size + 6pt`，这样更直观反映出两者的差别。

另外需要我们设计的就是 `text-alignment` 和 `first-line-indent`，即文字的对齐方式和段落的首行缩进，这在办公文档中也是需要根据公司规定进行修改的一点。

如果你对上一章还有印象，应该对 CSS 中选择器分组的方式有印象：

```
h1, h2, h3, h4, h5, h6 {
  margin: 30px 0 10px 0;
  color: #4a4a4a;
  line-height: 1.4;
}
```

通过这种方式，可以同时对一到六级标题进行设定。不过写起来还是太过繁琐，所以在 USS 中，设计了**通用选择器** `heading-all` 代替。它会作为所有标题的基础样式，然后再通过 `heading-1` 等选择器，单独为每级标题设定样式。类似的通用选择器还是有 `block-all`，`list-all`。

当有多个选择器对同一种元素起作用时，就会可能产生设定冲突问题。如果 `defaults` 中的字体是华文黑体，`heading-all` 中是苹方，最后的效果到底该是什么呢?

在 USS 中， 如果出现互相冲突的设置，将采取「**靠后原则**」—— **后出现的值将覆盖先出现的值**。`heading-all` 在 USS 中写在 `defaults` 后面，就以 `heading-all` 设置的字体为准。

还有一种类似的情况是一种元素出现在另一种元素中，但没有设定特定的后代或者子元素选择器，例如：

```
> # Heading with **strong text**
```

此时 `strong text` 将采取「**就近原则**」—— 离文字最近的语法标记对应的选择器先发挥作用。如果对一级标题、引用和粗体的设置为：

```
block-quote {
    font-family:    "Cochin"
    font-slant:     italic
}

heading-1 {
    font-family:    "Futura"
    font-size:      24pt
}

inline-strong {
    font-weight:    bold
}
```

最后对 `strong text` 这两个词的设定将是：

```
font-family:    "Futura"
font-slant:     italic
font-size:      24pt
font-weight:    bold
```

在这一小节中，我们以标题和段落的选择器为例，了解了 **USS 中的变量、表达式和通用选择器**的概念，它们让 USS 写起来更佳简便，在后面的内容中，还是经常看到它们的身影。

**设置行内元素、块级元素和列表**

同 HTML 一样，我们也可以把 DOCX 文档中区分出行内元素和块级元素。

前者有行内代码、粗体，后者常见的有引用块、代码块等等。

在设计 USS 时，设计行内元素和块级元素的思路是一致的：首先通过通用选择器 `block-all` 和 `inline-all` 来设定基本样式，再根据需求去修改。

那么具体有哪些元素需求我们设定呢？下面我将 Markdown XL 中的语法标记与 USS 中的选择器一一对应起来，供大家参考。

- 代码：inline–code
- 评论：Inline–comment
- 删除：delete
- 强调：inline–emphasis
- 链接：inline–link
- 重点：inline–mark
- 原始源：inline–raw
- 粗体：inline–Strong
- 代码块：block–code
- 注释块：block–quote

- 原始源块：block–raw
- 有序列表：list–ordered
- 无需列表：list–unorderd

可以看到，这里的选择器加上之前介绍的标题、图片，涵盖了除了视频外的语法标记，用法与其他选择器无异，这里不再一一赘述。不过这里再详细介绍其中的几处特别用法。

如果你在 Ulysses 中输入了原始源块，导出成 PDF 时会发现它们没有被一并导出，其中的玄机就在于：

```
block-raw {
    // kills raw source blocks
    visibility:             hidden
}
```

`block-raw` 选择器中，可见性 `visibility` 被设置成了隐藏。因为被设定为原始源块的内容，都不会导出。同理，行内的删除和评论也都是被设为了隐藏：

```
inline-delete {
    // deletes deletions
    visibility:             hidden
}

inline-comment {
    visibility:             hidden
}
```

如果你想将这些内容导出，将可见性设置为 `visible` 即可。不过与此同时，别忘了为这些内容设置合适的样式。

在为行内代码设定样式时，发现了一种新的语法：

```
inline-code : @code {
    style-title:            "Inline Code"
}
```

其实和之前介绍过的变量类似，`@code` 也可以看作变量，不过它对应的是一组声明。在 Papers 中，它被设定为：

```
@code {
    font-family:            "Courier"
    font-size:              10pt
    font-weight:            normal
    font-slant:             normal
}
```

在使用时，就不用再将这些属性再写一遍，直接使用 `@code` 来替代就好。

## 小结

本章的内容较多，我们**先是了解了排版流程 USS 的用法，再依次将 USS 中常见的选择器和术语做了介绍**。

然而限于篇幅，在一章中很难深入细致地介绍办公文档排版的方方面面，回顾本章的内容，希望你能了解到：

- 从 Ulysses 到 Word 之间，可以用 Ulysses 完成哪些排版任务；
- 一份 USS 文件该如何构建。

如果你能掌握这两部分内容，相信一份办公文档的样式文件设计对你来说不在话下。

经过这一章的介绍，可以说已经挖掘了 Markdown 配合 Ulysses 直接导出成 PDF 和 DOCX 文档的潜力，但是写过学术报告的人会明白，插入参考文献、数学公式甚至 LaTeX 内容，是学术文档中必不可少的。没有这些，很难说 Ulysses 能用于学术写作。

那么下一章，我们将对学术写作时出现的痛点，借助 Pandoc 等外部工具来解决。

我们下一章见。

1. 在官方文档中的英文名为 Style Setting，直译为样式设定。为了便于理解，这里按照 CSS 中文翻译的习惯，译为「属性」 。↩
2. Relative Selectors 没有官方以及通用的中文译名，为了方便大家阅读，我这里将它翻译为「关联选择器」↩
3. Ulysses 中的关联选择器没有特定的术语，我这里是根据 CSS 中各自对应的派生选择器命名。↩
4. 因为字体名间有空格，使用时注意写成 "ST Heiti"。↩